Cairo University
Faculty of Engineering
Computer Engineering
Fall 2018

Design and Analysis of Algorithms
Homework 3

DEADLINE Saturday, December 1st 2018, 11:59 PM

Your code should be taking into consideration the following requirements:

1. Implement your code in C++ under Linux.
2. Use the Python script for each problem to a) compile your source file, b) run the binary file, c) validate the output of the program.
3. Do not alter the Python file.
4. Do not alter the program arguments
5. Consult he Python script to determine the name of your main source file. The source file name is included in the Python file for compilation.
6. Consult the Python script to determine the order and type of arguments passed to your program.
7. Do not submit back the Python script. We will test with our own script that follows the same process in the supplied one.
8. The code will be judged according to consistent coding convention, code cleanliness, and documentation.
9. Remember! Plagiarism is not tolerated. Any sign of cheating or plagiarism will be graded as ZERO in this assignment and all other assignments.

Deliverables:

- A brief (one page is enough) report.pdf document with the answers for the text questions in each problem (if applicable).
- Source code for every problem in a folder with problem name following the convention "pi" where i is the problem number. For example "p1" is the folder name for the first problem, and so on.
- Create a root directory named "hw3" containing the subfolders of all problems. It should contain too the PDF file.
- The final hierarchy should be:
    - hw3
        - report.pdf
        - p1
            - …..cpp
        - p2
            - …..cpp
        - pi
            - …..cpp
        - pn
            - …..cpp
- Submit "hw3.zip" which is the compressed "hw3" folder.

Submission:

- Go to www.elearn.eng.cu.edu and login with your student account (the same you use in the student portal).
- If not already enrolled, enroll in the course with the key sent to your class representative.
- Submit all deliverables through the website by the indicated deadline.

Problems

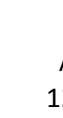
Problem 1:

Problem Statement:

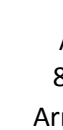
Given two binary search trees with distinct keys (containing n and m nodes with heights h1 and h2), return all common elements between them in O(n + m) time and O(h1 + h2) (heap+stack) space.

Input:

Examples of trees
Let the marker for NULL pointers be '-1'



Array Given : 13 12 -1 -1 -1



Array Given : 20 8 -1 -1 22 -1 -1

Array Given : 20 8 4 -1 -1 12 10 -1 -1 14 -1 -1 -1

Array Given: 20 10 8 5 -1 -1 -1 -1 -1

GIVEN Variables:

n: Number of pointers in the first trees (0 < n < 100000)
ai: array element of the first tree (-2 < ai < 10e9)
m: Number of pointer in the second trees (0 < m < 100000)
bi: array element of the second tree (-2 < bi < 10e9)

Input file format:

n
a1 a2 a3 .... an
m
b1 b2 b3 ... bm

Output:

Please first output the number of common elements **K** alone in a line then output the common elements in a new line.

Output file format:

k
c1 c2 c3 ..... ck

Problem 2:

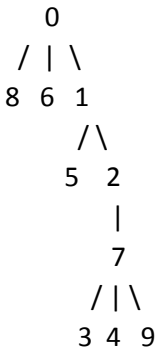
Problem Statement:

Given an array which has the indices of the parent of every node in a tree. Find the height of the tree. For the root node, the parent has index -1. Indices start from 0.

Note: Height of the Tree is zero when empty and else it is the number of nodes in the longest path.

Input:

Example: [-1 0 1 7 7 1 0 2 0 7]



GIVEN Variables:

n: Number of nodes in the trees (0 < n < 100000)

ai: array element of the tree (-2 < ai < 10e9)

Input file format:

n
a1 a2 a3 .. an

Output:

print one number in one line **H**

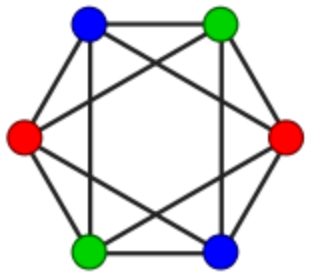
Output file format:

H

Problem 3:

Problem Statement:

Find a way of coloring the vertices of a graph such that no two adjacent vertices are colored using same color.



VERY IMPORTANT NOTE:

I want the sub optimal solution of this problem using a greedy algorithm. Please be careful and choose good criteria to design a greedy algorithm very near to the optimal solution of this problem. Otherwise, a poor criteria will result in a far from optimal solution.
Please don't do it brute force.
We can accept multiple solutions not the exact solution.

Input:

GIVEN Variables:

n: Number of Vertices in the Graph (0 < n < 1000)

m: Number of Edged in the Graph (0 < m < n(n-1)/2)

x , y: the node in the graph

Input file format:

n m
x1 y1
x2 y2
.
.
.
xm ym

Output:

Output every vertex with its color from 0 -> n-1 (Every vertex in one line)

Vertices are [0,n)

Colors are from 0 -> d

Notes:

Since d is maximum degree, a vertex cannot be attached to more than d vertices. When we color a vertex, at most d colors could have already been used by its adjacent. To color this vertex, we need to pick the smallest numbered color that is not used by the adjacent vertices. If colors are numbered like 0, 1, ..., then the value of such smallest number must be between 0 to d (Note that d numbers are already picked by adjacent vertices).

Output file format:

0 c1
1 c2
2 c3
.
.
n-1 cn

Problem 4:

Problem Statement:

Consider a row of n coins of values v1 . . . vn, where n is even. We play a game against an opponent by alternating turns. In each turn, a player selects either the first or last coin from the row, removes it from the row permanently, and receives the value of the coin. Determine the maximum possible amount of money we can definitely win if we move first.

Note: The opponent is as clever as the user.

Let us understand the problem with few examples:

1. 5, 3, 7, 10 : The user collects maximum value as 15(10 + 5)
2. 8, 15, 3, 7 : The user collects maximum value as 22(7 + 15)

Does choosing the best at each move give an optimal solution?
No. In the second example, this is how the game can finish:
1.
.....User chooses 8.
.....Opponent chooses 15.
.....User chooses 7.
.....Opponent chooses 3.
Total value collected by user is 15(8 + 7)
2.
.....User chooses 7.
.....Opponent chooses 8.
.....User chooses 15.
.....Opponent chooses 3.
Total value collected by user is 22(7 + 15)
So if the user follows the second game state, maximum value can be collected although the first move is not the best.

Input:

GIVEN Variables:

n: Length of the array (0 < n < 1000)

ai: array element (0 < ai < 10e9)

Input file format:

n
a1 a2 a3 .. an

Output:

Output the maximum possible amount of money we can definitely win if we move first.

ans: The integer required

Output file format:

ans

Problem 5:

Problem Statement:

Given a string, find the longest substring which is palindrome. For example, if the given string is “formoemennemeomfor”, the output should be “moemennemeom”.

Note:

Subarray/Substring

A subbarray is a contiguous part of array. An array that is inside another array. For example, consider the array [1, 2, 3, 4], There are 10 non-empty sub-arrays. The subbarays are (1), (2), (3), (4), (1,2), (2,3), (3,4), (1,2,3), (2,3,4) and (1,2,3,4). In general, for an array/string of size n, there are n*(n+1)/2 non-empty subarrays/subsrings.

Input:

GIVEN Variables:

n: The length of the given str (0 < n < 1000)

str: The actual string

Input file format:

18

formoemennemeomfor

Output:

Output the length of the longest palindromic substring in a line then output the actual substring in a new line

Output file format:

12

moemennemeom